**Designing of different digital and analog low pass and high pass filters and applying them of different singals.**

**Vinay Chittam**

**Student id: fet110**

**Department of Electrical Engineering**

**University of Texas at San Antonio**

**Instructor: Dr. Sos Agaian**

**Table of contents**

**Abstract**

**Introduction-----------------------------------------------------------------------------------------------1**

**Filters-------------------------------------------------------------------------------------------------------2**

**Low pass filters-------------------------------------------------------------------------------------------3**

**Circuit design of practical low pass filters----------------------------------------------------------3.1**

**Classification of filters----------------------------------------------------------------------------------3.2**

**Digital and analog parameters -----------------------------------------------------------------------3.3**

**Algorithm to design analog low pass filter ---------------------------------------------------------3.4**

**Algorithm to design digital low pass filter from the constructed analog filter--------------3.5**

**MATLAB simulation of first order analog low pass and high pass filters-------------------4**

**Matlab simulation to design digital 1st low pass filter using bi- linear transforms--------4.1**

**First order low pass filter on sound signal---------------------------------------------------------4.2**

**Design of Analog and digital High pass filters from low pass filters--------------------------5**

**MATLAB implementation of analog and digital high pass filters.----------------------------6**

**Digital Highpass 1st order filter from analog filter using bi-linear transform--------------6.1**

**Butter worth analog low pass filter------------------------------------------------------------------7**

**Design steps for analog and digital butter worth low pass filter ------------------------------8**

**Butter worth low pass filter on sound signal-------------------------------------------------------8.1**

**Design of digital Butterworth High pass filter from low pass butter worth filter----------9**

**Butter worth high pass filter on sound signal------------------------------------------------------9.1**

**Comparison of all the filters to find out the best one:--------------------------------------------10**

**Conclusion-------------------------------------------------------------------------------------------------11**

**References-------------------------------------------------------------------------------------------------12**

**Matlab code-----------------------------------------------------------------------------------------------13**

**Abstract**

The main purpose of this project is to understand the basic concepts and the characteristics of a filter which will help us analyze the functionality of filters on signals. In this project, I am going to design various kinds of digital filters (low pass and high pass), such as first order low pass filter, first order high pass filter, butter worth low pass filter and butter worth high pass filter. While designing these digital filters, I followed an algorithm which will be clearly explained later on in the project. In short, we first map the digital specifications to analog specifications and design analog filter, which is then transformed to digital filter by using a technique called “Bi-linear transform”.

I have followed this technique to design all the digital filters specified. To understand and to prove that the filter works, I applied them on various noisy signals. The filtered signal is compared with the original signal and the noisy signal. It was observed that the noise part was considerably removed from the signal. Therefore leaving the signal more reliable.

Finally, I have compared the functionality of all the filters. My simulation results show that, butter worth filters best approximates the ideal filter, because of its flat response in the pass band and stop band.

**1. Introduction**

In many applications, such as in communications, medical industries, stock market, machinery industries, bio medical industries, the generation of signals and faithful reconstruction of signals became very essential part for their functionality.

For example, in telecommunication, the user transmits the signal through the channel, the signal when reaches the receivers end should interpret what the user at the transmitter end conveys. This happens when the signal at the transmitter and receiver are the same, which is not always the case in real time applications. In real time environment, as the signal propagates through channel, it gets degraded due to noise. The noise is added to the signal due many factors, such as rain, obstacles, human beings etc. Thus the signal at the receiver end might lose valuable information. Thus a faithful reconstruction at the receiver end is very important for successful communication.

For this very purpose we use what is called as “Filters”.

**2. Filters**

Basically the role of filters is to extract the desired information from the signal. In telecommunication, the transmitted signal gets noisy as it travels along the channel. Since we do not desire to have noise component at the receiver, we can remove it by applying it to the filter.

The filter operates on the input signal and transforms it depending on the transform function of the filter. It is this transfer function of the filter that plays the most vital role. The transfer function of the filter convolves with the input signal to give the filtered output signal in time domain. Since convolution involves complicated calculation, we simplify the process by taking Fourier Transform of the input signal and the transfer function of the filter. Then the input signal and the transfer function of the filter are now in frequency domain, which are multiplied to get the filtered output in frequency domain. (Convolution in time domain is multiplication in frequency domain, proved in project 1 on images). This process also shows the importance of Fourier Transforms in filtering process.

The following block diagram shows the process of signal filtering.

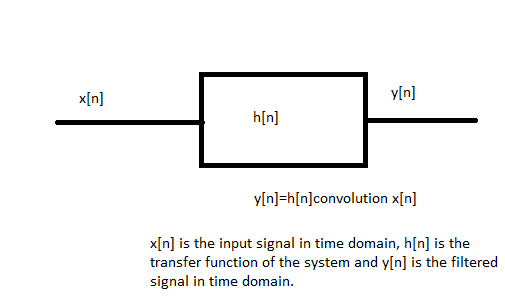


Figure 1

On application of Fourier Transform, the following changes happen:

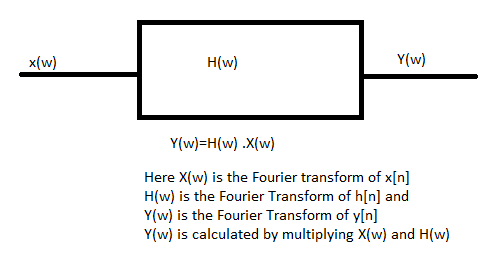


Figure 2

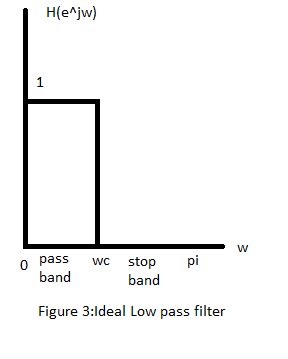
**Note: The system has to be Linear Time Invariant.**

The filters are classified into low pass filters, high pass filters, band pass filters and band stop filters. In this project I am going to design different kinds of low pass and high pass filters. Once low pass filter is designed all the other filters can be easily built around the constructed low pass filter. (This will be explained in detail in this report later on as it involves a lot of maths.)

**3. Low pass filters**

The filters that allows all the low frequency components to pass are called low pass filters. We can specify a point on the frequency scale called “cut off frequency” and all the frequency components that fall in this range are allowed to pass by the filter, while the components that are beyond the range of the cut off frequency are rejected by the filter. The range of frequency components allowed by the filter is called pass band range and the range of frequencies beyond the cut off frequency is called stop band range.

Ideally a filter looks as follows:



Here, the parameter Wc is the cut off frequency, the magnitude of the low pass filter is 1 only for the range 0 to Wc. The magnitude is 0, for the range Wc to pi.

The low pass filter works as follows:

First the signal in time domain is taken and Fourier Transform is applied to it. The resultant frequency domain signal is multiplied with the transfer function of the filter “H(w)”. If the entire signal is within the range 0 to Wc, the signal is multiplied with 1, which gives the same signal. This means that if the signal is a low frequency signal i.e. if it lies in the range 0 to Wc, the entire signal is allowed to pass by the filter.

If we have a high frequency signal, whose frequency is more that Wc, then the filter rejects it, by multiplying the signal with 0. Thus the high frequency signals are attenuated by the low pass filter. This is the working of the ideal low pass filter.

But in reality it is not possible to design an ideal low pass filter. This can be noticed by taking the inverse Fourier Transform of the transfer function of the filter “H(W)”. The result is a sinc function with harmonics extending infinity in the time domain. In practice it is impossible to deal with infinity. Therefore we need to approximate the ideal low pass filter with a low pass filter that doesn’t have problems of dealing with infinity in time domain.

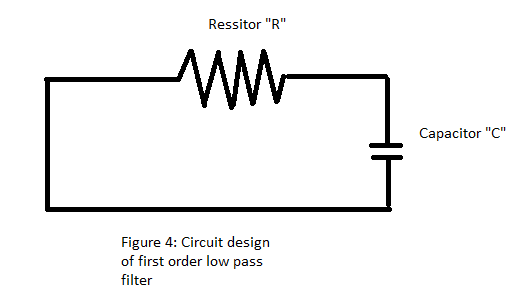
**3.1 Circuit design of practical low pass filters**

Since the ideal low pass filter is impossible to realize, practical low pass filters are to be created. Low pass filters are designed by a simple RC circuit by connecting a resistor in series with a capacitance. The input signal is sent through the resistor and the output is calculated across the capacitor.

The problem with the ideal filter is that since there is an abrupt jump at the cut off freuqncy Wc i.e. the magnitude falls abruptly from 1 to 0 at Wc, which is hard to realize.

If we design the filter using RC circuit and measure the output at the capacitor the magnitude will have a smoother roll off factor, rather than a sudden jump. This is because of the capacitor used, which is used to store the charge of the signal and discharges slowly. A capacitor takes some time to discharge, hence we get a smooth roll off factor as the charge decreases. If the number of capacitors used is 1, then it’s called a First order low pass filter. By using more capacitors i.e. increasing the order of the filter we can approximate to the ideal filter more closely.

The circuit design of a low pass filter is shown as follows:



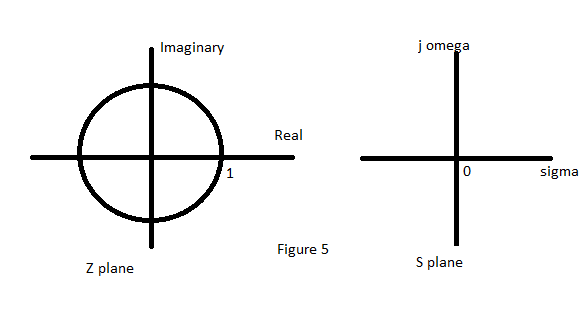
**3.2. Classification of filters**

Filters are classified into types, Analog filters and Digital filters. In this project, I am going to design both analog and digital filters. Analog filters are designed in rectangular coordinate system i.e. s-plane (s= sigma + j ohm, where ohm is analog frequency) called as analog/Laplace domain. Whereas the digital filters are designed in Z domain (z=e^jw, where w= digital frequency). In order to design a digital filter we first use bi-linear transformation technique, which is used to map an analog filter/analog specifications to digital filter/digital specifications. To design digital filters for a given set of digital specifications, we first map the specifications to analog specifications and then build analog filter. This is done because analog filters have been around for much longer and it’s easy to build digital filters by using analog filters. The built analog filter can be transformed to digital filter by using “Bi- linear transformation technique”. The bi- linear technique will also be explained in detail later on in this report.

The digital filters are again classified into two types. 1. IIR and 2. FIR

IIR stands for Infinite impulse response. If the impulse response/transfer function of the filter is infinite then the filter is called as infinite impulse response filter. If the impulse response is finite then the filter is called as finite impulse response filter. The ideal filter that we have seen earlier can be either approximated by IIR or FIR.

The s and z domain are shown are shown in the figure below:



By using bi linear transformation we can map filter in s domain to z domain, in order to construct digital filter from analog filter.

If the frequency in s plane is on the vertical line i.e. when s=j ohm then it is mapped to z plane right on the circle whose radius is 1.

If the frequency is on the left hand side of the s plane, then it is mapped inside the unit circle in z plane.

If the frequency is one the right hand side of the s plane, then it is mapped outside the unit circle in z plane.

This can be verified as follows: z= 1+s/1-s and the magnitude of z is given as mod(z^2)=mod ([1+s/1-s][1+s/1-s]\*]).

If we substitute s= j Ohm then magnitude of z becomes 1. In order to map any point on s domain to z domain we have to substitute s=sigma + j ohm in the magnitude equation. By substituting so, will prove that frequencies on the left hand side will be inside the unit circle and frequencies on the right hand side of s plane will be outside the unit circle and when s=j ohm the frequencies will be mapped on to the unit circle.

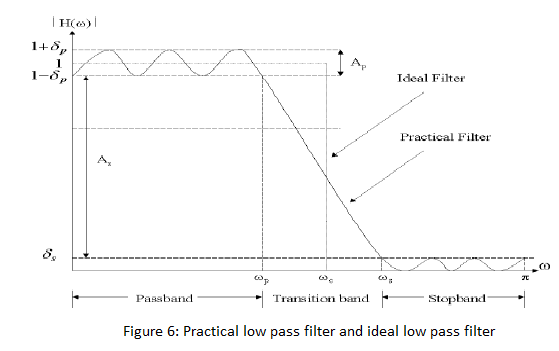
**3.3 Digital and analog parameters**

W(omega ) is called as digital frequency

Ohm is called as analog frequency.

The above two parameters, digital and analog frequencies are related as W/2=tan ^-1 ohm and W= 2 tan^-1 ohm (Bi-liner Transform) . By using these two parameters we can convert digital frequencies to analog and vice versa.

There are some other parameters used for designing a low pass filter, which can be better understood by viewing the following image.



We know that the magnitude of ideal filter is 1. The practical filter tries to approximate the ideal filter, the better approximation leads to better filter. In the practical filter, the magnitude is not exactly 1, it varies around 1 from 1-delta p to 1+delta p, where delta p is the error by which the magnitude is away from the ideal magnitude 1.

Similarly, the magnitude of the ideal low pass filter is 0 at the stop band region, but in practical filter the magnitude varies around 0 to delta s.

Ap is the attenuation at the pass band frequency “Wp”. In ideal filter, in the pass band the filter should not attenuate the signal, but in practical filter we still have some attenuation given by Ap. Lesser the attenuation “Ap”, the better the filter as it approximates closely to the ideal filter.

“As” is attenuation at the stop band frequency “Ws”. In ideal filter the signal will get attenuated totally, but in the practical case there will not be full attenuation because of the ripples in the stop band as shown in the above figure. Greater the attenuation “As”, better the filter, as it approximates closely to the ideal filter.

Wc is the cut off frequency at which the power or magnitude of the signal is 3db or 1/sqrt 2. This is because the capacitor discharges to a point where the response of the filter is reduced to 1/sqrt 2 times the amplitude of signal.

Delta W is the transition band which starts from Wp and ends at Ws. For the ideal filter transition band will be 0. But practically, transition band exits because of the capacitor, as explained earlier.

The attenuation As, Ap and Ac can be calculated using the following formulas:

As=-20log10(delta s) dB; Ap=20log10(1+delta p/1-delta p)dB

From the above equations we can calculate delta p and delta s as follows: delta p=10^Ap/20 -1/10^Ap/20+1 and delta s=10^-As/20.

Ac= 3dB in dB scale or in linear scale Gc=1/sqrt 2= gc^2=1/2. Also Wc or W can be calculated from frequency of the signal as Wc=2pifc/fs or W=2pif/fs, where fs is the sampling frequency.

With these set of specifications, we can design a filter.

**3.4 Algorithm to design analog and digital low pass filter**

With some given set of specifications (f, fc, Ac) or (W,Wc, Gc), Ac=3dB or Gc= 1/ sqrt2.

We can map these digital specifications to analog specifications using the formulas

**ohm( analog frequency)=tan W/2 and ohm c= tan Wc/2. ---------(1)**

And by using one of the well known analog filter designs we can know the transfer function of the first order filter given as follows

**H(s)= alpha/s + alpha------------(2)**

, where the parameter alpha can be calculated from the above mentioned specifications as follows:

Substitute s=j ohm, then the transfer function becomes **H(j ohm)= alpha / j ohm+ alpha.**

The magnitude square function is given as **mod(H(j ohm))^2=alpha^2/ohm^2+ alpha^2.**

In terms of cut off frequency the equation becomes **mod (H(j ohm c))^2= alpha^2/ohm c^2+alpha^2= Gc^2,** solving this equation gives us the value of alpha, which is

**alpha= Gc/sqrt(1-Gc^2) .ohm c.----------(3)**

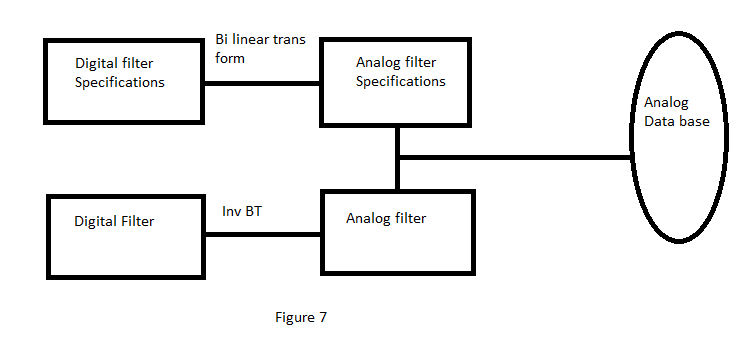
By substituting the value of alpha and s=j ohm in the equation (2), we get the transfer function of the analog filter. Now we can filter the signal.

**Steps to filter:**

1. First find the Fourier Transform of the signal
2. Generate noise and find the Fourier Transform of the noise signal.
3. Add noise signal to the original signal to get a noisy signal.
4. Multiply the noise signal to the transfer function of the signal to get the filtered signal.
5. It will be observed that the noise present in the signal above the cut off frequency will be eliminated by the filter.

**3.5 Algorithm to design digital low pass filter from the constructed analog filter**

The same steps are followed as mentioned above, i.e. converting digital specifications to analog specifications using the equations mentioned above. After obtaining the analog specification, we design the analog transfer function or analog filter. From this we map it to digital domain by using bi- linear transform to get digital filter. This process can be understood with following block diagram:



After designing the analog filter, to map it to digital domain, we just have substitute “s=1-z^-1/1+z^-1” in the transfer function of analog filter, to get the transfer function of digital filter. Thus we can construct the digital filter from the analog filter using bi linear transformation technique.

**4. MATLAB simulation of first order analog low pass and high pass filters**

For implementing analog low pass filter, I have taken the following specifications:

Sinusoid varying at 1 cycle per second so that we can clearly see the working of low pass.

The amplitude of sinusoid is 10V

Noise amplitude is 0.2 times the original amplitude

Transfer function of analog 1st order filter=H(s)=alpha/alpha+s

By following the steps mentioned the filter can be designed using MATLAB software.

****

**Figure 8: First order Analog filter with cut off 0.5**

**Note: Initially I have taken my cut off at 0.5hz, where as the noise ranges from 0.4 hz to 0.5 hz. If it was an ideal filter even at cut off frequency 0.3hz, we could have filtered the noise. But since this a practical filter, at cut off frequency the magnitude reduces to 1/2 times the amplitude of the original amplitude. Even at 0.5 hz cut off frequency, we will have some roll off factor which will not reject the noise. So adjust the cut off frequency such that noise is eliminated. In order to do so, we obviously have to choose cut off further on the left side of the noise range. In the next figure I have selected 0.2hz as my cut off to eliminate noise.**

**Figure 8.1: First order Analog filter with cut off 0.2hz**

**Note: For the cut off frequency 0.2hz, the noise is mostly eliminated. Thus the signal is filtered.**

It can be noticed that, if we choose our cut off such that the noise part lies outside the range, the filter will reject the noisy part from the spectrum. Thus the low pass filter only selects desired frequency components (low frequency comopents) depending on what the cut off frequency is.

**4.1 Matlab simulation to design digital 1st low pass filter using bi- linear transforms**

* The digital filter can be obtained by performing bi- linear transform.
* The digital filter can be built from the analog filter. After transforming the specifications to analog domain and building the analog filter, we have to substitute **Z= 1-Z^-1/1+Z^-1** in the transfer function of analog filter **H(s)=alpha /s+alpha**.
* The alpha also changes to **Alpha=(TanWc/2).Gc/sqrt(1-Gc^2)**
* With this transformation technique the digital filter can be designed from the analog filter.

The following figure shows the design and implementation of digital filters on signals.

****

**Figure 9: First order digital low pass filter**

**Note: At cut off frequency 50hz, the noise still exists as the range of noise extends from 40 to 60hz. Low pass filter eliminates only the frequency components within the range 0 to cut off. Therefore if I changed the cut off to 20hz, so that the filter can reject the noise which lies at high frequency.**

****

**Note: If I increase the cut off frequency to 20hz, all the noise components that lie in the range from 40 to 60hz will be removed by the low pass filter. The figure shows that the filter works perfectly file. The original signal is reconstructed.**

**4.2 First order low pass filter on sound signal**

For testing the low pass first order filter on the sound signal, I have chosen a sound file and subjected it to the filter designed. I have first added noise to the sound signal and then filtered it by using low pass filter. The sound files for original sound, noisy + original and filtered sound are presented to play in the presentation. The following figure demonstrates the process of filtering the sound signal.

****

**Note: The noise lies at around 80hz, so I have chosen the frequency 1hz to completely eliminate noise signal. The cut off can varied to see what happens at different frequencies. At cut off 1hz, I could remove the noise and reconstruct the signal.**

**5. Design of Analog and digital High pass filters from low pass filters**

High pass filters are used to allow only high frequency components and rejects all the low frequency components. It allows all the frequency components that lie above the cut off frequency and rejects the frequencies lying in the range from 0 to cut off frequency. Therefore for high pass filters, the pass band lies above the cut off frequency and stop band lies below the cut off frequency. For an ideal high pass filter the magnitude is 1 at the range outside the Wc (cut off frequency) and the magnitude of 0 in the range 0 to Wc. But practical high pass filter doesn’t have the exact same parameters as the ideal filter**. In practical high pass filter the response gradually increases to 1, instead of an abrupt jump because the capacitor now charges, i.e. stores the charge of the signal, that is the working of practical high pass filter**. The parameters mentioned for low pass are similar to parameters used for high pass.

It can be easily constructed from the low pass filter. As seen in the construction of low pass filter, the transfer function for analog filter was H(s)= alpha /alpha +s. In fact high pass can be constructed by subtracting low pass filter from 1. i.e. **high pass= 1- low pass.**

The transfer function of high pass will now be **H(s)= s/alpha +s.**

Since the transfer function has changed, the parameter alpha also changes, i.e.

**alpha =sqrt(1-Gc^2)/Gc. Tan ohm c.**

Thus the analog high pass filter can be constructed.

In order to design digital high pass filter, we can use bi linear transform to map from s domain to z domain. By substituting **s=1-z^-1/1+z^-1** in the transfer function of high pass filter we get digital filter.

**6. MATLAB implementation of analog and digital high pass filters.**

For implementing analog high pass filter, I have taken the following specifications:

Sinusoid varying at 1 cycle per second so that we can clearly see the working of low pass.

The amplitude of sinusoid is 10V

Noise amplitude is 0.2 times the original amplitude

Transfer function of analog high pass 1st order filter H(s)=s/s+alpha

Alpha = ohmc.\*sqrt(1-Gcsq)./sqrt(Gcsq);

Gcsq=1/2(at cut off)

By following the steps mentioned the filter can be designed using MATLAB software.

****

**Figure 10 Analog high pass filter**

**Note: It can be noticed that, if we choose cut off frequency such that the noise lies outside the range of the cut off frequency , the filter allows the noise, since it is in the pass band region. I chose the cut off at 1hz, to test the working of the filter. At 1hz, the filter should allow the noise to pass, because the noise lies above the cut off frequency 1hz. The figure shows clearly that the noise is allowed by the filter.**

**In the next figure, I have chosen frequency such that it lies in the stop band so that the filter can reject the noise components. For this purpose, I kept the cut off frequency as 5hz.**

****

**Note: If I increase my cut off frequency to 5hz, the noise will be in the cut off range , and therefore the filter allows only the components above the cut off frequency. Thus the noise is eliminated.**

**6.1 Digital Highpass 1st order filter from analog filter using bi-linear transform.**

* For the digital filter I just substituted s=(1-z^-1)/(1+z^-1) in the transfer function of analog high pass filter i.e. H(s)= s/s+alpha. This will transform into z domain H(z).
* Alpha value also changes .
* Alpha = tan(wc/2).\*sqrt(1-Gcsq)./sqrt(Gcsq);
* First I chose cut off to be 50hz. Therefore all the frequency components above the cut off will be allowed to pass by the filter. Even the noise will be passed.

****

**Note: If the cut off is chosen such that the noise lies above the cut off or pass band, the filter passes the noise components. Here the cut off is 50 hz, and noise is in the range 45 to60hz, so the noise is allowed. In the next figure, I selected my cut off to be 80hz, to check if the filter rejects the noise.**

****

**Figure 11 Digital high pass filter**

**If I increase the cut off such that the noise is within stop band, the filter rejects the noise. Here the cut off is above 60hz, i.e. 80hz and the filter successfully rejects the noise from the signal.**

**Frist order digital high pass on sound signal:**

For testing the high pass first order filter on the sound signal, I have chosen a sound file and subjected it to the filter designed. I have first added noise to the sound signal and then filtered it by using high pass filter. The sound files for original sound, noisy + original and filtered sound are presented to play in the presentation. The following figure demonstrates the process of filtering the sound signal. 

**Note: The noise spectrum lies at 80hz to 120 hz. By choosing the cut off frequency 230hz totally away from the cut off range i.e. by making the noise signal sit in the stop band region, we can eliminate the noise component from the sound signal. By playing the original sound signal, noisy signal and the filtered signal we can differentiate that the filtered signal indeed has filtered the noise.**

**7. Butter worth analog low pass filter**

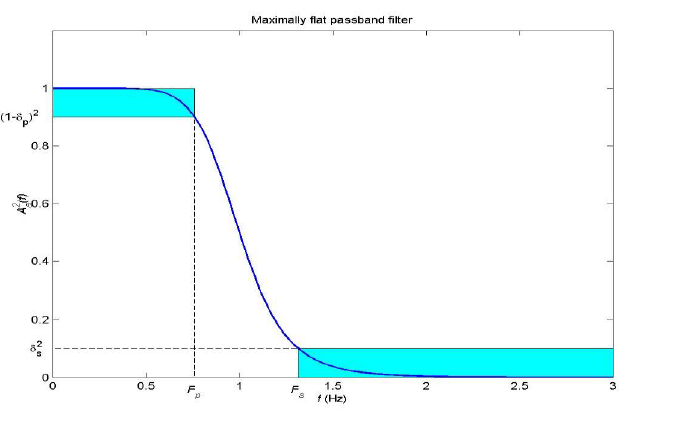
Butter worth is an IIR filter, also known as maximal flat response filter. It is called so because the magnitude response of this filter is flat without any ripples in the pass band and stop band. As the order of the filter increases the magnitude response becomes flatter and flatter, approximating the ideal filter much closely.

The magnitude response of an analog Butter worth filter is given as

**Mod(H(j ohm))= 1/[1+(ohm/ohm c)^2N]^1/2--------(1)**

Here, N is the order of the filter, ohm is the analog frequency of the signal and ohm c is the analog cut off frequency of the signal.

The magnitude response is shown below:

****

**Figure 12 Butter worth magnitude response**

**From Dr. Sos Agaians lecture notes.**

It can be seen that the magnitude response doesn’t have any ripples in the pass band and stop band like that of a typical low pass filter. The magnitude is 1 at 0 ohms and at cut off frequency the magnitude reduces to 3db or 1/sqrt 2.

As the order increases the roll off factor decreases, which makes the filter looks almost similar to ideal low pass filter.

To derive the transfer function of a stable system, we substitute ohm= s/j which we get from from s =j ohm.

For the sake of convenience normalize the cut off frequency to ohm c= 1. Now the transfer function in equation (1) becomes **1/ 1+(s/j)^2N or 1/1+(ohm)^2N---------------(2)**

Therefore **H(s)H(-s)= 1/1+(-s^2)^N-------(3)**

By equating the denominator to 0, we get all the poles of the system. For the system to be stable the poles have to be on the left hand side of the s plane.

The transfer function of an analog Butter worth high order filter can be designed by the values of the poles that lie on the left hand side of s plane.

Therefore it’s necessary to find a formula to calculate the poles that lie on the left hand side of the s plane so that our filter is stable. The stable poles can be calculated by the following formula:

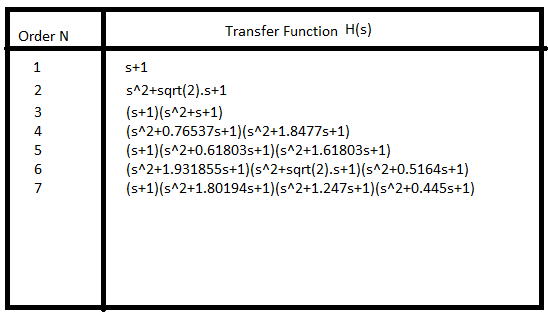
**poles = pi/2+(2k-1)pi/eN,----------(4)**

where N is the order and k varies from 1 to N

By using this formula for order 1, i.e. N=1 gives us one pole.

For N=2 we get 2 poles and so on.

By using these poles the transfer function of the filter can be found out. The list of transfer functions for various values of N ranging 1 to 7 are tabulated as follows:

****

**Figure 13: List of transfer functions for different orders.**

**Note: The above transfer functions were derived for normalized cut off frequencies i.e. ohm c = 1. In order to find the transfer functions for a particular cut off frequency, we just have to substitute s=s/ohm c in the transfer function H(s).**

The same parameters defined for a typical low pass filters are used in butter worth filter as well. From a given set of specifications {Ohm pass, Ohm stop, Ap, As}, the order of the filter can be calculated by using the following formula:

**N>=log(sqrt(10^0.1As -1/10^0.1Ap -1))/log (ohm stop/ohm pass)------------(5)**

From the above given specifications, we can even calculate the cut off frequency, which is given by the formula: **ohm c=ohm pass/(10^0.1 Ap -1)^1/2N----------------------(6)**, which gives the cut off frequency for the given specifications.

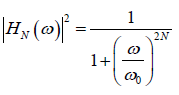
**8. Design steps for analog and digital butter worth low pass filter**

* From the given specifications, calculate the order N using the equation (5). If the specifications are in digital format, convert them using bi –linear transform as explained earlier for low pass filter.
* Depending on what order we get choose the corresponding transfer functions which are tabulated above in figure 13.
* For the given specifications calculate cut off frequency using equation (6).
* After finding the cut off frequency substitute s= s/cut off frequency in the transfer function. Thus the analog filter is built.
* From this analog filter a digital filter is easy to construct. We just have to substitute s=1-z^-1/1+z^-1 into the transfer function H(s) to get digital filter transfer function. Thus the bi- linear transform function helps us design digital filter from the analog filter.

**8. MATLAB implementation of digital Butterworth low pass filter.**

By following the steps mentioned above we can design the butter worth filter in matlab software. In this project I implemented the butter worth filter using the general formula for magnitude response **H(W)= sqrt(1/1+(W/Wc)^2N ).**

I used the following specifications to design the filter:

* Sinusoid wave varying at 1 cycle/sec (1hz)
* Amplitude 10v
* Amplitude of noise .2\*10v
* Magnitude response
* 
* Cut off freq=45hz
* N is the order. N=1

The following figure shows the simulation of the filter**.**

****

**Figure 14 Digital Butterworth low pass filter**

**Note: For order N=1, the roll off factor of the filter is longer. The cut off frequency is 45hz, exactly at the point where the noise spectrum starts. Ideally the noise has to be removed, i.e. if the filter is an ideal filter. As discussed, the butter worth is called as maximally flat response filter and best approximates the ideal filter. So it should remove the noise, but it doesn’t because the filter order is 1. The response gets flatter only for higher order filter. Since it’s a low order filter the roll off factor is still long. Therefore the noise will be passed by the filter.**

**Now I will prove that the butter worth closely approximates the ideal filter at higher orders. For this purpose, I chose the same cut off frequency i.e. 45hz at which the noise spectrum starts. The only difference is that now I changed the order form N=1 to N=10. The results are as follows:**

****

**Note: It can be seen that the magnitude response is flatter at the pass band and stop band. Since the magnitude response is flat at the stop band, the noise is almost all removed. And also the roll off factor decreased significantly, trying to look close like an ideal filter which has an abrupt jump at cut off frequency. The noise is removed for order N=10 at the same cut off frequency used before for order N=1. This proves that, the butter worth filter closely approximates the ideal filter for higher order.**

Now I chose my order to be N=20, this will even more improve the performance of the filter by making the magnitude response more flatter at pass and stop band and also by reducing the roll off factor.



**Note: For order N=20, the signal becomes evenmore smoother, since the roll off decreased much more. As the order increases, the roll off becomes almost vertical straight line like that of an ideal filter.**

**8.1 Butter worth low pass filter on sound signal**

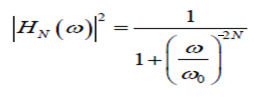
For testing the butter worth low pass filter on the sound signal, I have chosen a sound file and subjected it to the filter designed. I have first added noise to the sound signal and then filtered it by using butter worth low pass filter. The sound files for original sound, noisy + original and filtered sound are presented to play in the presentation. The following figure demonstrates the process of filtering the sound signal.

****

**Note: By choosing the cut off frequency 10hz, I could completely remove the noise components from the original sound signal. The cut off frequency can be varied to see different effects of noise. By keeping the noise components in the stop band we can eliminate the noise.**

**An important thing to notice in butterworh filter : I have chosen the order to be N=5 therefore the roll off factor is somewhat vertical. The roll off factor is also variable. If we choose the order to be N=1, the roll off will long therefore little noise might be included in the pass band.**

**9. Design of digital Butterworth High pass filter from low pass butter worth filter**

The butterworth high pass filter rejects, high frequency components which are above the cut off frequency. It is easy to design the butter worth high pass filter since we have already constructed a butter worth low pass filter. After constructing a low pass butter worth filter we just have to subtract low pass from 1. i.e. High pass filter= 1- low pass filter. This will give us the magnitude response of the high pass filter as follow: 

From this new magnitude response we can design the butter worth high pass filter.



**Note: At cut off frequency, 57hz where the noise spectrum range exactly ends, the ideal high pass filter should reject the noise since it lies in stop band i.e. below the cut off. But this doesn’t happen because the order is N=1, for which the roll factor is longer. If the roll off factor is sharp then the filter would have been like an ideal filter and would have rejected the noise. In the next figure, I have increased the order of the filter by keeping the cut off constant i.e. 57hz.**

****

**Note: Since the order is now N=10, the roll off factor decreased and the signal has become smoother due to elimination of noise. The noise in the stop band is removed by the filter and also the magnitude response is flatter in the pass band and stop band.**

**9.1. Butter worth high pass filter on sound signal**

For testing the butter worth high pass filter on the sound signal, I have chosen a sound file and subjected it to the filter designed. I have first added noise to the sound signal and then filtered it by using butter worth high pass filter. The sound files for original sound, noisy + original and filtered sound are presented to play in the presentation. The following figure demonstrates the process of filtering the sound signal.



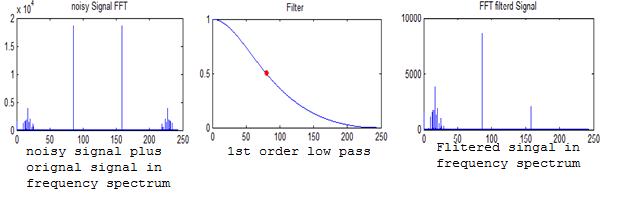
Note: **By choosing the cut off frequency 220hz, I could completely remove the noise components from the original sound signal. The cut off frequency can be varied to see different effects of noise. By keeping the noise components in the stop band we can eliminate the noise.**

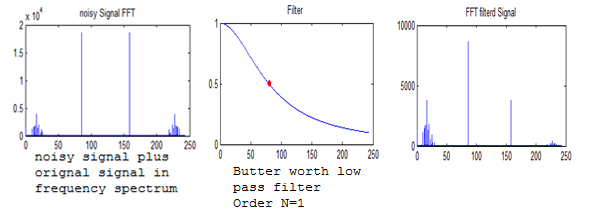
**An important thing to notice in butterworh filter : I have chosen the order to be N=10 therefore the roll off factor is somewhat vertical. The roll off factor is also variable. If we choose the order to be N=1, the roll off will long therefore little noise might be included in the pass band.**

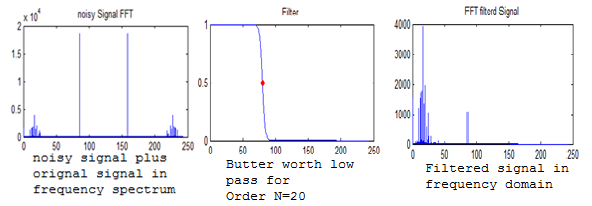
**10. Comparison of all the filters to find out the best one:**

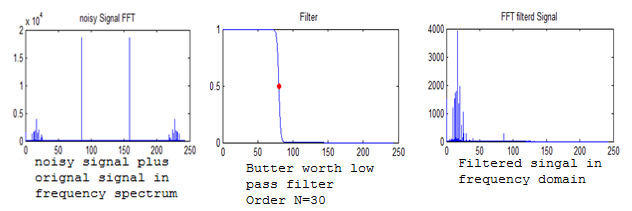
**In order evaluate and compare which filter gives the best response, I have created an unbiased platform by making the specifications constant and checking all the filters for these specifications. Comparison can be easily done by listening, so I first considered sound signal.**

**The best way to compare is to check how closely the filter approximates the ideal filter. As discussed earlier, the ideal low pass filter, eliminates the frequencies just above the cut off frequency. For this purpose, I chose the cut off at 80hz, exactly from where the noise spectrum starts. For this specification, I tested all the filters designed. Obviously the filter which removes the noise totally is the best filter. Here are the simulation results.**

****

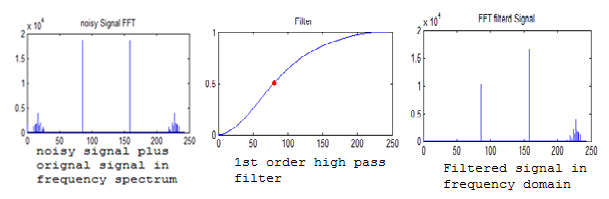
****

****

****

**From the above simulation, it can be seen that, butter worth filter gives the best results provided the order is higher.**

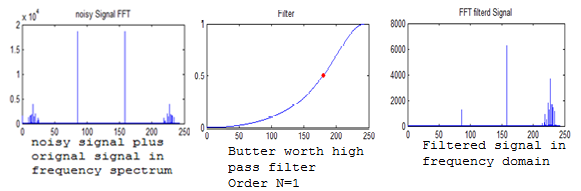
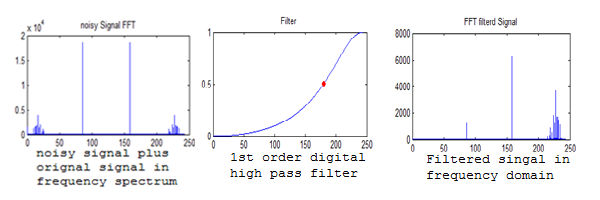
**Now for the same cut off frequency if we use high pass filter, obviously by definition the noise remains because it lies in the pass band of the high pass filter. I’m going to show it by using 1st order high pass filter for same cut off frequency. It should allow the noise to pass.**

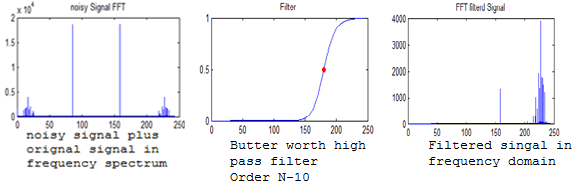
****

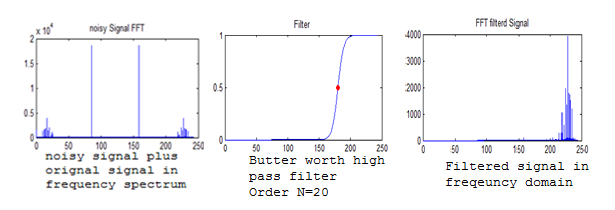
**Note: The above figure shows that the noise has not been removed. This is quite obvious because for high pass to work, we need to keep the cut off higher, as the high pass only allows high frequency components. This will be the case for all the high pass filters.**

**To compare and check which high pass is best:**

**For high pass we need to choose high frequency point as the cut off so that the noise is removed. So, I have chosen the cut off at 180hz right at the point where noise spectrum ends. Ideal high pass has to remove the noise as it has least roll off factor. Let us check which filter is best i.e. which filter approximates the ideal high pass filter.**

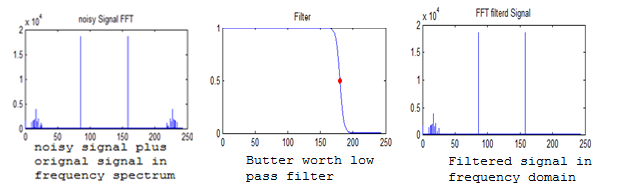
****

****

****

**Note: From the above figure, it can be noticed that a normal high pass 1st order filter bad when compared to butter worth filter. The order 1 butter worth and normal high pass filter look almost same. But as the order increases the butter filter gets better and better. As seen in the figure at order 10, there is little noise. At order 20 the noise is totally removed.**

**If I compare these high pass filters with low pass filter obviously, I will get a lot noise, because for that same cut off 180hz, the noise lies in the pass band of the low pass filter. The following example shows it.**

****

**Note: Obviously a low pass filter will allow noise if the cut off is 180hz.**

**11. Conclusion**

Thus I have designed 1st order low pass and high pass filters, both analog and digital using bi- linear transformation. Also from the low pass, I have designed high pass filters, both analog and digital.

I also have designed a butter worth filter both low pass and high pass.

It was observed form the simulation results that the butter worth filter gives the best response as the order increases. Also the name “Maximally Flat Response Filter” was justified as the filter behaved almost similar to ideal filter.

**12. References**

* Dr. Sos Agaian’s Lecture notes
* DSP text book by P. Ramesh Babu

**13. Matlab Code**

1. 1st order analog low pass filter on sinusoid

%

% Filter

%

clear;clc;close all;

n=100;

A=10;

t = linspace(0,2\*pi,n);

s = A\*sin(2\*pi\*1\*t);

nois = 0.2\*A\*sin(2\*pi\*1000\*t);

% nois = 0.2\*A\*rand(1,n);

% nois = wgn(1,n,1);

sn = s + nois;

% sn =awgn(s,10,'measured');

snfft = (fft(sn));

%% Filter design and implement

w = linspace(0,1\*pi/2,n);

%wc = pi/1.5;

ohmc=0.5;

ohm=tan(w/2);

Ac = 3;

Gc = 10^(-Ac/20);

Gcsq = Gc.^2;

alph = ohmc.\*sqrt(Gcsq)./sqrt(1-Gcsq);

Hw = alph./(1i\*ohm + alph);

filtrd = snfft.\*abs(Hw).^2;

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot(ohm,(abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot(ohm,(abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot(ohm,(abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(ohm,abs(Hw).^2);title('Filter ');

Hwc = interp1(ohm,Hw,ohmc);

hold on; plot(ohmc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(ohm,abs(filtrd));title('FFT filterd Signal');

subplot(339);

filtrd\_rec = ifft(((filtrd)));

m = max(s);

recA = max(real(filtrd\_rec));

plot(t,m\*real(filtrd\_rec)./recA);title('Filterd Signal');axis([0 t(end) -A A]);

2. 1st order digital low pass filter using bi linear transform

%

% Filter

%

clear;clc;close all;

%%

% initial values and signals

n=100;

A=10;

t = linspace(0,2\*pi,n);

s = A\*sin(2\*pi\*1\*t);

nois = 0.2\*A\*sin(2\*pi\*1000\*t);

% nois = 0.2\*A\*rand(1,n);

% nois = wgn(1,n,1);

sn = s + nois;

snfft = (fft(sn));

%% Filter design and implementation

w = linspace(0,1\*pi,n);

fc = 20;

wc = pi\*fc/n;

Ac = 3; % Gcsq = 1.5

Gc = 10^(-Ac/20);

Gcsq = Gc.^2;

alph = tan(wc/2).\*sqrt(Gcsq)./sqrt(1-Gcsq);

b0 = alph/(1+alph);

b1 = b0;

a1 = -(1-alph)/(1+alph);

z = exp(1i\*w);

Hw = (b0 + b1\*(z).^(-1))./(1+a1\*(z).^(-1));

filtrd = snfft.\*abs(Hw).^2;

%% Plot data

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot((abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot((abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot((abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(abs(Hw).^2);title('Filter ');

Hwc = interp1(w,Hw,wc);

hold on; plot(fc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(abs(filtrd));title('FFT filterd Signal');

subplot(339);

filtrd\_rec = ifft(((filtrd)));

m = max(abs(s));

recA = max(real(filtrd\_rec));

plot(m\*real(filtrd\_rec)./recA);title('Filterd Signal');

3. 1st order low pass filter on sound signal.

%

% Filter

%

clear;clc;close all;

%% load data

fname = 'HBD.wav';

[s,fs]=audioread(fname);

s=s';

[r,n]=size(s);

A = max(s);

tl = n/fs;

t = linspace(0,tl,n);

disp('Playing orignal signal....');

sound(s,fs);

%% add noise

% nois = wgn(1,n,-30); % -30db wgn

nois = 0.2\*A\*sin(2\*pi\*1e6\*t); % sinusoidal noise ...

sn = s + nois;

pause(12);

disp('Playing noisy signal');

sound(sn,fs);

audiowrite(['Noisy\_' fname],sn,fs);

%% define filter and filter params

scl = 1000;

xf = linspace(0,n/scl,n); % scalling X axis of frequency by 1000 (kHz)

fc=1;

w = linspace(0,1\*pi,n);

wc = pi\*fc\*scl/n;

Ac = 3;

Gc = 10^(-Ac/20);

Gcsq = Gc.^2;

alph = tan(wc/2).\*sqrt(Gcsq)./sqrt(1-Gcsq);

b0 = alph/(1+alph);

b1 = b0;

a1 = -(1-alph)/(1+alph);

z = exp(1i\*w);

Hw\_t = (b0 + b1\*(z).^(-1))./(1+a1\*(z).^(-1));

Hw = repmat(Hw\_t,[r,1]);

%% FFT os signals and filter implementation

snfft = ((fft(sn)));

sfft = ((fft(s)));

noisfft = ((fft(nois)));

filtrdfft = snfft.\*abs(Hw).^2;

filtrd\_rec = ifft((filtrdfft));

% Hw = ftrans2(Hw\_t);

%%

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot(xf,(abs((sfft))));title('FFT Orignal Signal');

subplot(335);

plot(xf,(abs((noisfft))));title('FFT noise Signal');

subplot(336);

plot(xf,(abs(((snfft)))));title('noisy Signal FFT');

subplot(337);

plot(xf,abs(Hw).^2);title('Filter ');

Hwc = interp1(w,Hw,wc);

hold on; plot(fc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(xf,abs(filtrdfft));title('FFT filterd Signal');

subplot(339);

m = (max(s));

recA = max(real(filtrd\_rec));

fil\_final = m\*real(filtrd\_rec)/recA; % applying gain...

plot(t,fil\_final);title('Filterd Signal');axis([0 t(end) -A A]);

pause(12);

disp('Playing recovered signal');

sound(fil\_final,fs);

audiowrite(['Lowpass\_filterd' fname], fil\_final,fs);

4. 1st order analog high pass filter

%

% Filter

%

clear;clc;close all;

%% load data

n=100;

A=10;

t = linspace(0,2\*pi,n);

fm=1;

s = A\*sin(2\*pi\*fm\*t); % sampling frequency sould be more than 4kHz

[r,n]=size(s);

%% add noise

nois = 0.3\*A\*sin(2\*pi\*1000\*t);

% nois = wgn(1,n,0);

sn = s + nois;

%% define filter and filter params

w = linspace(0,.99\*pi,n);

ohm=tan(w/2);

ohmc=5;

Ac = 3; % Ac value close to 3 gives Gcsq = 0.5

Gc = 10^(-Ac/20);

Gcsq = Gc.^2;

alph = ohmc.\*sqrt(1-Gcsq)./sqrt(Gcsq);

S=1i\*ohm;

Hw = S./(S + alph);

%% FFT os signals and filter implementation

snfft = ((fft(sn)));

sfft = ((fft(s)));

noisfft = ((fft(nois)));

filtrdfft = snfft.\*abs(Hw).^2;

filtrd\_rec = ifft((filtrdfft));

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot(ohm,(abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot(ohm,(abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot(ohm,(abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(ohm,abs(Hw).^2);title('Filter ');

Hwc = interp1(ohm,Hw,ohmc);

hold on; plot(ohmc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(ohm,abs(filtrdfft));title('FFT filterd Signal');

subplot(339);

m = max(s);

recA = max(real(filtrd\_rec));

plot(t,m\*real(filtrd\_rec)./recA);title('Filterd Signal');axis([0 t(end) -A A]);

5. 1st order digital high pass filter using bi- linear technique

%

% Filter

%

clear;clc;close all;

%% load data

n=100;

A=10;

t = linspace(0,2\*pi,n);

fm=1;

s = A\*sin(2\*pi\*fm\*t); %

[r,n]=size(s);

%% add noise

% nois = wgn(1,n,0);

nois = 0.2\*A\*sin(2\*pi\*1000\*t);

sn = s + nois;

%% define filter and filter params

fc=80; %

w = linspace(0,1\*pi,n);

wc = pi\*fc/n;

Ac = 3; % Ac value close to 3 gives Gcsq = 0.5

Gc = 10^(-Ac/20);

Gcsq = Gc.^2;

alph = tan(wc/2).\*sqrt(1-Gcsq)./sqrt(Gcsq);

b0 = 1/(1+alph);

b1 = -b0;

a1 = -(1-alph)/(1+alph);

z = exp(-1i\*w);

Hw = (b0 + b1\*(z).^(-1))./(1+a1\*(z).^(-1));

%% FFT os signals and filter implementation

snfft = ((fft(sn)));

sfft = ((fft(s)));

noisfft = ((fft(nois)));

filtrdfft = snfft.\*abs(Hw).^2;

filtrd\_rec = ifft((filtrdfft));

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot((abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot((abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot((abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(abs(Hw).^2);title('Filter ');

Hwc = interp1(w,Hw,wc);

hold on; plot(fc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(abs(filtrdfft));title('FFT filterd Signal');

subplot(339);

m = max(abs(s));

recA = max(real(filtrd\_rec));

plot(t,m\*real(filtrd\_rec)./recA);title('Filterd Signal');axis([0 t(end) -A A]);

6. 1st order High pass filter on sound

%

% Filter

%

clear;clc;close all;

%% load data

fname = 'HBD.wav';

[s,fs]=audioread(fname);

s = s';

[r,n]=size(s);

A=max(s);

t = linspace(0,n/fs,n);

disp('Playing orignal signal....');

sound(s,fs);

%% add noise

% nois = wgn(1,n,0);

nois = 0.2\*A\*sin(2\*pi\*1e6\*t);

sn = s + nois;

% pause(12);

disp('Playing noisy signal');

% sound(sn,fs);

audiowrite(['Noisy\_' fname],sn,fs);

%% define filter and filter params

scl = 1000;

xf = linspace(0,n/scl,n); % scalling X axis of frequency by 1000 (kHz)

fc=230;

w = linspace(0,1\*pi,n);

wc = pi\*fc\*scl/n;

Ac = 3; % Ac value close to 3 gives Gcsq = 0.5

Gc = 10^(-Ac/20);

Gcsq = Gc.^2;

alph = tan(wc/2).\*sqrt(1-Gcsq)./sqrt(Gcsq);

b0 = 1/(1+alph);

b1 = -b0;

a1 = -(1-alph)/(1+alph);

z = exp(-1i\*w);

Hw = (b0 + b1\*(z).^(-1))./(1+a1\*(z).^(-1));

%% FFT os signals and filter implementation

snfft = ((fft(sn)));

sfft = ((fft(s)));

noisfft = ((fft(nois)));

filtrdfft = snfft.\*abs(Hw).^2;

filtrd\_rec = ifft((filtrdfft));

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot(xf,(abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot(xf,(abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot(xf,(abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(xf,abs(Hw).^2);title('Filter ');

Hwc = interp1(w,Hw,wc);

hold on; plot(fc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(xf,abs(filtrdfft));title('FFT filterd Signal');

subplot(339);

m = max(abs(s));

recA = max(real(filtrd\_rec));

fil\_final = m\*real(filtrd\_rec)./recA;

plot(t,fil\_final);title('Filterd Signal');axis([0 t(end) -A A]);

pause(12);

disp('Playing recovered signal');

sound(fil\_final,fs);

audiowrite(['Highpass\_filterd' fname], fil\_final,fs);

7. Butter worth low pass filter on sinusoid

%

% Filter

%

clear;clc;close all;

%% read data add noise

n=100;

A=10;

t = linspace(0,2\*pi,n);

s = A\*sin(2\*pi\*1\*t);

nois = 0.2\*A\*sin(2\*pi\*1000\*t);

% nois = 0.2\*A\*rand(1,n);

% nois = wgn(1,n,1);

sn = s + nois;

snfft = (fft(sn));

%% Filter

w = linspace(0,1\*pi,n);

fc = 45; % on the scale of 100 where n =100

wc = fc\*pi/n;

N = 20; % filter order

Hw = sqrt(1./(1+(w/wc).^(2\*N)));

filtrd = snfft.\*abs(Hw).^2;

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot((abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot((abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot((abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(abs(Hw).^2);title('Filter ');

Hwc = interp1(w,Hw,wc);

hold on; plot(fc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(abs(filtrd));title('FFT filterd Signal');

subplot(339);

filtrd\_rec = ifft(((filtrd)));

m = max(s);

recA = max(real(filtrd\_rec));

fil\_final = m\*real(filtrd\_rec)/recA;

plot(t,fil\_final);title('Filterd Signal');axis([0 t(end) -A A]);

9. Butter worth low pass filter on sound signal

%

% Filter

%

clear;clc;close all;

%% read data

fname = 'HBD.wav';

[s,fs]=audioread(fname);

s=s';

A = max(s);

n = size(s,2);

t = linspace(0,n/fs,n);

disp('Playing orignal signal....');

sound(s,fs);

%% add noise

nois = 0.2\*A\*sin(2\*pi\*1e6\*t);

% nois = 0.3\*A\*rand(1,n);

sn = s + nois;

pause(12);

disp('Playing noisy Signal');

audiowrite(['Noisy\_' fname],sn,fs);

sound(sn,fs);

snfft = (fft(sn));

%% Filter

scl = 1000;

xf = linspace(0,n/scl,n); % scalling X axis of frequency by 1000 (kHz)

fc = 10; %

w = linspace(0,1\*pi,n);

wc = pi\*fc\*scl/n;

N = 5; % filter order

Hw = sqrt(1./(1+(w/wc).^(2\*N)));

filtrd = snfft.\*abs(Hw).^2;

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot(xf,(abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot(xf,(abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot(xf,(abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(xf,abs(Hw).^2);title('Filter ');

Hwc = interp1(w,Hw,wc);

hold on; plot(fc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(xf,abs(filtrd));title('FFT filterd Signal');

subplot(339);

filtrd\_rec = ifft(((filtrd)));

m = max(s);

recA = max(real(filtrd\_rec));

fil\_final = m\*real(filtrd\_rec)/recA;

plot(t,fil\_final);title('Filterd Signal');axis([0 t(end) -A A]);

pause(12);

disp('Playing recovered Signal');

sound(fil\_final,fs);

audiowrite(['buterworth\_lowpass\_filterd' fname], fil\_final,fs);

10. Butter worth high pass on sinusoid

%

% Filter

%

clear;clc;close all;

%% read data add noise

n=100;

A=10;

t = linspace(0,2\*pi,n);

s = A\*sin(2\*pi\*1\*t);

nois = 0.2\*A\*sin(2\*pi\*1000\*t);

% nois = 0.2\*A\*rand(1,n);

% nois = wgn(1,n,1);

sn = s + nois;

snfft = (fft(sn));

%% Filter

w = linspace(0,1\*pi,n);

fc = 57; % on the scale of 100 where n =100

wc = fc\*pi/n;

N = 10; % filter order

Hw = sqrt(1./(1+(w/wc).^(-2\*N)));

filtrd = snfft.\*abs(Hw).^2;

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot((abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot((abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot((abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(abs(Hw).^2);title('Filter ');

Hwc = interp1(w,Hw,wc);

hold on; plot(fc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(abs(filtrd));title('FFT filterd Signal');

subplot(339);

filtrd\_rec = ifft(((filtrd)));

m = max(s);

recA = max(real(filtrd\_rec));

fil\_final = m\*real(filtrd\_rec)/recA;

plot(t,fil\_final);title('Filterd Signal');axis([0 t(end) -A A]);

11. Butter worth high pass filter on sound

%

% Filter

%

clear;clc;close all;

%% read data

fname = 'HBD.wav';

[s,fs]=audioread(fname);

s=s';

A = max(s);

n = size(s,2);

t = linspace(0,n/fs,n);

disp('Playing orignal signal....');

sound(s,fs);

%% add noise

nois = 0.2\*A\*sin(2\*pi\*1e6\*t);

% nois = 0.3\*A\*rand(1,n);

sn = s + nois;

pause(12);

disp('Playing noisy Signal');

audiowrite(['Noisy\_' fname],sn,fs);

sound(sn,fs);

snfft = (fft(sn));

%% Filter

scl = 1000;

xf = linspace(0,n/scl,n); % scalling X axis of frequency by 1000 (kHz)

fc = 220; %

w = linspace(0,1\*pi,n);

wc = pi\*fc\*scl/n;

N = 10; % filter order

Hw = sqrt(1./(1+(w/wc).^(-2\*N)));

filtrd = snfft.\*abs(Hw).^2;

figure;

subplot(331);

plot(t,s);title('Orignal Signal');axis([0 t(end) -A A]);

subplot(332);

plot(t,nois);title('noise Signal');axis([0 t(end) -A A]);

subplot(333);

plot(t,sn);title('Noisy signal');axis([0 t(end) -A A]);

subplot(334);

plot(xf,(abs((fft(s)))));title('FFT Orignal Signal');

subplot(335);

plot(xf,(abs((fft(nois)))));title('FFT noise Signal');

subplot(336);

plot(xf,(abs((snfft))));title('noisy Signal FFT');

subplot(337);

plot(xf,abs(Hw).^2);title('Filter ');

Hwc = interp1(w,Hw,wc);

hold on; plot(fc,abs(Hwc).^2,'ro','markerfacecolor','r','markersize',5);

subplot(338);

plot(xf,abs(filtrd));title('FFT filterd Signal');

subplot(339);

filtrd\_rec = ifft(((filtrd)));

m = max(s);

recA = max(real(filtrd\_rec));

fil\_final = m\*real(filtrd\_rec)/recA;

plot(t,fil\_final);title('Filterd Signal');axis([0 t(end) -A A]);

pause(12);

disp('Playing recovered Signal');

sound(fil\_final,fs);

audiowrite(['buterworth\_highpass\_filterd' fname], fil\_final,fs);